CLAIMS

What is claimed is:

1. A method for automatically generating a description of a data exchange format based on computer program source code expressed in a source language, the method comprising the computer-implemented steps of:

receiving, from a source code file, comment data including first data indicating a parameter of the data exchange format, wherein the comment data is ignored by a source code processor of the source language;

receiving from the source code file second data, associated with the comment data, indicating a statement that defines a class of data objects in the source language; and

automatically generating, based on the first data and the second data, third data that describes the data exchange format.

2. A method as recited in Claim 1, further comprising generating, based on the first data and the second data, a module to convert a data object of the class of data objects into a data item of the data exchange format as described by the third data.

3. A method as recited in Claim 1, further comprising generating, based on the third data, a module to convert a data object of the class of data objects into a data item of the data exchange format as described by the third data.

4. A method as recited in Claim 1, further comprising generating, based on the first data and the second data, a module to derive a data object of the class of data objects from a data item of the data exchange format as described by the second data.

5. A method as recited in Claim 1, further comprising generating, based on the third data, a module to derive a data object of the class of data objects from a data item of the data exchange format as described by the third data.

6. A method as recited in Claim 1, wherein the third data is formatted according to a database query language.

7. A method as recited in Claim 1, wherein the third data is formatted according to a symbolic markup language.

8. A method as recited in Claim 1, wherein the third data is formatted according to extensible markup language (XML).

9. A method as recited in Claim 1, wherein the third data comprises one or more statements in an XML schema document.

10. A method as recited in Claim 1, wherein the third data is one or more statements in an XML document type definition (DTD) document.

11. A method as recited in Claim 1, wherein the third data is one or more statements in an XML document type definition (DTD) document, and wherein the parameter is at least one of a root element associated with an entire DTD document, an element and an attribute of an element.

12. A method as recited in Claim 1, wherein the third data is one or more statements in an XML document type definition (DTD) document, and wherein the first data includes one or more properties of the parameter.

13. A method as recited in Claim 1, wherein the source language is Java®.

14. A method as recited in Claim 1, wherein the source language is Java®, and wherein the first data includes a tag for an automated Java documentation system.

15. A method as recited in Claim 1, wherein the source language is Java®, wherein the first data includes a tag for an automated Java documentation system, and wherein the tag is a user-defined tag for the Java documentation system.

16. A method as recited in Claim 1, wherein the source language is Java®, wherein the first data includes a tag for an automated Java documentation system, wherein the tag is a user-defined tag for the Java documentation system, and wherein said step of generating the third data is performed by a user-defined routine invoked by the automated Java documentation system in response to the tag.

17. A method for binding a data exchange format with an application having source code in a particular language, the method comprising the steps of:

inserting, into the source code within comment data ignored according to a processor for the particular language, first data including a tag and a parameter of a data exchange format; and

causing a processor to produce second data for configuring the data exchange format based at least in part on the first data.

18. A method as recited in Claim 17, further comprising the steps of:

causing the processor to produce at least one of a module for marshaling data objects into a data item in the data exchange format as configured by the second data and a module for de-marshaling data objects from a data item in the data exchange format as configured by the second data; and

building the application based on the source code and at least one of the module for marshaling data and the module for de-marshaling data.

19. A method as recited in Claim 18, wherein the particular language is the Java language, wherein the tag is a user-defined tag of an automated Java documentation system; and wherein said step of causing a processor to produce the second data further comprises providing a routine, invoked by the automated Java documentation system in response to the tag, to produce the second data.

20. A method as recited in Claim 19, wherein the particular language is the Java language; wherein the tag is a user-defined tag of an automated Java documentation system; and wherein said step of causing a processor to produce at least one of the module for marshaling and the module for de-marshaling further comprises providing a routine, invoked by the automated Java documentation system in response to the tag, to produce at least one of a Java module for marshaling and a Java module for de-marshaling.

21. A computer-readable medium carrying one or more sequences of instructions for binding a data exchange format with an application having source code in a particular language, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

receiving, from a particular file that includes the source code, comment data including first data indicating a parameter of the data exchange format, wherein the comment data is ignored by a source code processor of the particular language;

receiving from the particular file second data, associated with the comment data, indicating a statement that defines a class of data objects in the particular language; and

generating, based on the first data and second data, third data for configuring the data exchange format.

22. An apparatus for binding a data exchange format with an application having source code in a particular language, comprising:

means for receiving, from a particular file that includes the source code, comment data including first data indicating a parameter of the data exchange format, wherein the comment data is ignored by a source code processor of the particular language;

means for receiving from the particular file second data, associated with the comment data, indicating a statement that defines a class of data objects in the particular language; and

means for generating, based on the first data and second data, third data for configuring the data exchange format.

23. An apparatus for binding a data exchange format with an application having source code in a particular language, comprising:

a processor;

one or more stored sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of:

receiving, from a particular file that includes the source code, comment data including first data indicating a parameter of a data exchange format, wherein the comment data is ignored by a source code processor of the particular language;

receiving from the particular file second data, associated with the comment data, indicating a statement that defines a class of data objects in the particular language; and

generating, based on the first data and second data, third data for configuring the data exchange format.